1 Objectives

In this lecture we extend the idea of local state from last time to create a simple implementation of objects, and discuss its limitations. We will also show the message dispatch model of objects, which allows for inheritance and virtual functions.

Your objectives:

- Be able to explain what an object is.
- Know how to implement an object using records and HOFs.
- Know how to implement an object using a message dispatcher.
- Be able compare the record and dispatcher models.
- Major goal 1: be able to simulate objects in a language lacking them.
- Major goal 2: understand how objects work "under the hood".

2 Examples

What is an object?

- Data and functions are grouped together.
- Functions have their own local state.
- Objects can send and receive *messages*.
- Objects can refer to *themselves*.

Using recursive records.

```
let mkPoint newloc =
  let rec this =
  { loc = ref newloc;
    getx = (fun () -> pi1 !(this.loc));
    gety = (fun () -> pi2 !(this.loc));
    draw = (fun () -> report !(this.loc));
    move = (fun dl ->
              this.loc := movept !(this.loc) dl)}
 in this;;
```

Using a dispatcher.

```
let mkPoint x y =
  let x = ref x in
  let y = ref y in
  fun st ->
    match st with
    | "getx" -> (fun _ -> !x)
    | "gety" -> (fun _ -> !y)
    | "movx" -> (fun nx -> x := !x + nx; nx)
    | "movy" -> (fun ny -> y := !y + ny; ny)
    |  _ -> raise (Failure "Unknown message.")
```